**استخراج روابط از متن فارسی با روش های داده کاوی**

**Relation extraction from Persian text using data mining methods**

1. **مقدمه**
2. **معرفی استخراج اطلاعات[[1]](#footnote-1) و استخراج اطلاعات آزاد[[2]](#footnote-2)**

استخراج اطلاعات روشی برای بدست آوردن دانش ساخت یافته ساده از متن است. هدف از این سیستم ها یافتن و درک بخش های مرتبط بهم در متن به صورت محدود است. بسیاری از این سیستم ها، به استخراج اطلاعات از درون تکه های متن می­پردازند و یک نمایش ساختیافته از اطلاعات مرتبط بهم در متن را ارائه می­کنند. از اطلاعات بدست آمده در این سیستمها می­توان برای تولید پایگاه دانش بهره برد. این سیستم هادو هدف عمده را دنبال می­کنند:

* سازمان دهی اطلاعات داخل متن به نحوی کارا برای استفاده.
* قرار دادن اطلاعات در یک فرم دقیق منطقی که اجازه استنتاج و بررسی بیشتر را به الگوریتم ها و روش های کامپیوتری دیگر می­دهد.

بسیاری از الگوریتم های پیشرفته پردازش متن نمی­توانند با متن ورودی به صورت خام کار کنند و نیازمند دریافت برخی از اطلاعات به صورت ساختیافته هستند. مثلا به برچسب های مقوله دستوری[[3]](#footnote-3)یا اسامی موجود در متن یا نوع ارتباط آنها با هم در متن، نیازمندند.

سیستم های استخراج اطلاعات معمولا اطلاعاتی که به صورت حقایق آشکار[[4]](#footnote-4) است را از متن بیرون می­کشند. اطلاعاتی نظیر «چه کسی در چه زمانی چه کاری را با چه کسی انجام داد؟».به عنوان نمونه درآمد یک نفر یا شرکت، دارایی های او، اعضای هیئت مدیره، دفاتر یک شرکت، دفتر مرکزی، مدیر عامل و استخراج هرگونه حقایق از این دست که از درون گزارش ها می­توان بیرون کشید. یا اسخراج کتاب های یک نویسنده، کارگردان یک فیلم سینمایی، محصولات یک کمپانی تولید خودرو یا نام مخترع یک وسیله با پردازش متون موجود در اینترنت. به عنوان نمونه به جمله زیر توجه کنید.

*اداره کل آموزش و پرورش و مراکز دفتری استانهای آن، در خیابان طالقانی تهران قرار دارند.*

ایده این است که سیستم بتواند جمله بالا را پردازش کرده و از داخل آن متوجه شود که «اداره کل آموزش و پرورش» و «مراکز دفتری استانها» نام دو سازمان است و به آنها برچسب ORG بدهد همچنین متوجه شود که این جمله درباره مراکز دفتری صحبت میکند و محل قرار گیری آنها را تعیین نموده است. بنابراین خروجی زیر را تولید کند:

|  |  |  |
| --- | --- | --- |
|  | Located(«اداره کل آموزش و پرورش», «خیابان طالقانی تهران»)  Located(«مراکز دفتری استانهای آموزش و پرورش», «خیابان طالقانی تهران») |  |

استخراج اطلاعات همچنین در یافتن اطلاعات از متون پزشکی-زیست شناسی مانند روابط بین دارو های مختلف و بیماری ها، تاثیراتی ژنتیکی داروهای مختلف، تاثیر ژن در بیماری ها، اسامی گونه های مختلف گیاهان و ... مورد استفاده فراوانی قرار گرفته است.

روش های سنتی برای تشخیص هر رابطه در متن، یک استخراج گر مخصوص آن آموزش می­دادند (Moldovan, 1993) (Soderland, 1999). این نوع روشها قابلیت گسترش به پیکره هایی که در آنها تعداد روابط هدف زیاد است، یا اینکه نوع روابط قابل پیشگویی از قبل نیست، نمی­باشند (Anthony Fader, 2011). استخراج اطلاعات آزاد این مشکل را با در نظر گرفتن رابطه به صورت عباراتی که در زبان انگلیسی بیانگر رابطه هستند حل کرد (Banko, 2007). در این روش خود عبارت رابطه نیز به صورت اتوماتیک تشخیص داده می­شود و در واقع مسئله از حالت دو متغییری (تعیین دو آرگومان رابطه) به سه متغییری (تعیین رابطه و دو آرگومان آن) تغییر یافت. تشخیص اتوماتیک عبارت رابطه امکان استخراج روابط اختیاری را از جملات می­دهد که نیاز به محدود شدن به یک فرهنگ لغات را حذف کرد. این روش ها توانستند موفقیت های خوبی را بر روی پیکره های کلان مستخرج از اینترنت و ویکی­پدیا بدست آورند (Weld, 2010) (Banko, 2007) (Jun Zhu, 2009). از خروجی سیستم های Open IE برای کمک به یادگیری ترجیحات لغوی[[5]](#footnote-5) (Alan Ritter, 2010)، استخراج دانش عرفی[[6]](#footnote-6) (Lin, 2010) و استلزام از متن[[7]](#footnote-7) (Schoenmackers, 2010) (Berant, 2011) استفاده شده است. همچنین از اطلاعات مستخرج از این سیستمها برای تولید هستان شناسی[[8]](#footnote-8) استفاده شده است (Stephen Soderland, 2010).

**2-1- بررسی نمونه سیستم های موجود**

اگر با دید کاربردی به استخراج اطلاعات نگاه کنیم؛ این علم هم اکنون در بسیاری از تکنولوژی های امروزی موجود بوده و حتی فراگیرشده است. در بسیاری از کاربردها مانند سرویس ایمیل گوگل یا اپل یا اندیس گذاری اینترنتی. همانطور که در تصویر زیر مشاهده میکنید، سرویس ایمیل اپل به صورت اتوماتیک تاریخ و ساعت هایی که درون متن باشد را تشخیص می­دهد و این امکان را به کاربر می­دهدکه تنها با یک کلیک آن را در تقویم خود برای یادآوری ذخیره کند.



تصویر 1 کاربرد استخراج اطلاعات در گوشی اپل

مثالی دیگر در جستجوی گوگل است. اگر به دنبال مقر اصلی یک شرکت چند ملیتی بگردیم؛ گوگل همراه با بازگرداندن نتایج جستجو، بهترین حدس خود را برای پاسخ به سوال ما، در ابتدا لیست نمایش می­دهد. به تصویر زیر توجه کنید.



تصویر 2 کاربرد استخراج اطلاعات در جستجوی گوگل

در ادامه برخی از سایت ها و پروژه های موجود در زمینه استخراج اطلاعات برای زبان انگلیسی معرفی می­شوند. پروژه Open Calais از خبرگزاری Reuter’s یک نمونه جالب از استخراج اطلاعات است. در این سایت بخشی به شکل دموی آنلاین وجود دارد که می­توان یک متن انگلیسی دلخواه را در آن قرار داد. در شکل زیر نمونه خروجی آن را مشاهده میکنید.







تصویر 3 دانش استخراج شده از متن توسط سایت Open Calais از خبرگزاری Reuter's

همچنین با قرار دادن کرسر موس بر روی هر کدام از موجودیت های نامدار، دانش استخراج شده از متن برای آن نمایش داده میشود. شکل زیر.



تصویر 4 شناسایی موجودیت نامدار. سایت Open Calais

پروژه AlchemyAPI برای 8 زبان انگلیسی، فرانسه، آلمانی، ایتالیایی، پرتغالی، روسی، اسپانیایی و سوئدی قابلیت استخراج موجودیت های نامدار را دارد. در شکل زیر نمونه ای از خروجی این سیستم را مشاهده می­کنید. متن ورودی، همان متن وارد شده در مثال بالا است.



تصویر 5 نمونه استخراج اطلاعات از متن توسط سایت AlchemyApi

از دیگر وب سایت ها میتوان به Evri و Yahoo’s term extraction اشاره نمود.

1. **معرفی زیر شاخه های استخراج اطلاعات**

**3-1- شناسایی موجودیت های نامدار[[9]](#footnote-9)**

یک زیر کلاس بسیار مهم از استخراج اطلاعات، شناسایی موجودیت نامداراست که در آن، به دنبال یافتن و دسته بندی موجودیت های نامدار (اسامی) موجود در متن هستیم. بنابراین قدم اول یافتن اسامی است. در واقع محدوده اسامی در متن را تعیین می­کنیم.

*سرانجام در سومین روز هفدهمین دوره مسابقات آسیایی در اینچئون کره ‌جنوبی، طلسم بی ‌مدالی کاروان ایران شکسته شد و نجمه خدمتی، تیرانداز هجده ‌ساله تیم ملی کشور مان با درخشش خیره ‌کننده ‌ای به مقام قهرمانی آسیا رسید و نخستین طلای این رقابت‌ها را به نام کشورمان سند زد تا فراموش کنیم، ‌تیم فوتبال امید کشورمان امروز ‌رسما از رقابت‌های آسیایی اینچئون حذف شد .*

قدم بعدی دسته بندی اسامی یافته شده است.

*سرانجام در سومین روز هفدهمین دوره مسابقات آسیایی در اینچئون کره‌جنوبی، طلسم بی ‌مدالی کاروان ایران شکسته شد و نجمه خدمتی، تیرانداز هجده ‌ساله تیم ملی کشورمان با درخشش خیره ‌کننده‌ای به مقام قهرمانی آسیا رسید و نخستین طلای این رقابت‌ها را به نام کشورمان سند زد تا فراموش کنیم، ‌تیم فوتبال امید کشورمان امروز ‌رسما از رقابت‌های آسیایی اینچئون حذف شد.*

قرمز: سازمان –سبز: شهر –آبی: کشور –نارنجی: نام فرد–بنفش: قاره–سبز تیره: تاریخ

موجودیت[[10]](#footnote-10) به چیزی خیالی یا واقعی گفته می‌شود که به ذات خویش وجود دارد. به عنوان مثال هر انسان یک موجودیت است، دانشگاه شهید بهشتی، کشور ایران یا شرکت مخابرات ایران یک موجودیت هستند. اما هوا یا خاک موجودیت نیستند. ایده پشت موجودیت نامدار این است که برخی از موجودیت ها دارای نام هستند. مثلا من محمود راحت نام دارم، کشورم ایران نام دارد، شهرم تهران نام دارد اما صندلی که روی آن نشسته ام با این حال که یک موجودیت است؛ اما نام ندارد. البته بحث اینکه چه چیزهایی موجودیت نامدار هستند و چه چیزهایی نیستند، کمی فلسفی است و دیدگاه های مختلفی در این زمینه وجود دارد. اما در این بحث ما به دنبال اسامی و موجودیت های نامداری هستیم که به راحتی قابل تمیز باشند و از نظر فلسفی روی موجودیت بودن آنها شکی نیست. همچنین در پردازش متن چیزهایی مثل تاریخ (یازده خرداد سال 1364) و زمان (ساعت ده شب –11:30) و مقدار و نام مواد شیمیایی، پروتئنی یا زیستی نیز به عنوان موجودیت های نامدار در نظر گرفته می­شوند. با این حال که آنها به هیچ عنوان موجودیت نامدار نیستند ولی در زمان تعیین اسامی نامدار در متن عموما این اسامی نیز در نظر گرفته می­شوند.

در زیر برخی از کاربرد های شناسایی موجودیت های نامدار را مشاهده می­کنید:

* از موجودیت های نامدار در متن می­توان برای اندیس گذاری یا لینک دادن استفاده کرد. (بسیاری از شرکت ها از روشهای مختلف برای استخراج اسامی در متن استفاده می­کنند و در صفحات وب خود به آنها لینک می­زنند. مثلا در صفحات وب بایولوژی نام پروتئین ها و ژن های متن به صورت لینک بوده و به صفحه ای که به تبیین آن موجودیت می­پردازد لینک شده اند.)
* در کاربرد تشخیص نظر[[11]](#footnote-11) کاربران، نیازمندیم که بدانیم نظر نویسنده متن در مورد یک موضوع خاص مثبت یا منفی است. علاوه بر آن نیازمندیم بدانیم این نظر در مورد چه کسی یا چه چیزی است. لذا استخراج موجودیت های نامدار (شامل نام شرکت و نام محصول) می­تواند به تشخیص درست نظر کاربران کمک نماید.
* اگر به شناسایی موجودیت های نامدار به عنوان پیش نیاز استخراج اطلاعات نگاه کنیم، کاربردهای زیادی برای آن وجود دارد. در استخراج اطلاعات بیشتر کاری که انجام می­گیرد، استخراج موجودیت های نامدار و سپس تعیین روابط بین آنها است.
* کاربرد در سیستم های پرسش و پاسخ. در این سیستم ها پاسخ بسیاری از انواع سوالات، موجودیت های نامدارند. اینکه یک چیز چه زمانی اتفاق افتاده یا چه کسی آن را انجام داده است. بنابراین اگر یک سیستم خوب استخراج موجودیت نامدار برای زبان داشته باشیم، کمک شایانی به پاسخ سوالات به صورت اتوماتیک می­کند.

**3-2- رفع ابهام مرجع گروه اسمی**

**3-3- استخراج روابط موجود در متن**

1. **بررسی روشها موجود**

کار روی استخراج خودکار اطلاعات از متن به دهه 1970 میلادی باز می­گردد (Etzioni, 2011). روش های اولیه با استفاده از قوانین دست ساز و بر روی یک ژانر و موضوع خاص کار می­کردند. به عنوان مثال سیستم JASPER در سال 1980 برای استخراج اطلاعات مالی از روی اخبار رویترز به وجود آمد. گسترش این روش زمان بر و خطا خیز بود. در دهه 1990 دسته جدیدی از سیستم های استخراج اطلاعات ظهور یافتند. که در آنها به جای تولید دستی قوانین، سیستم به صورت خودکار قوانین را از روی مجموعه منتخب از جملات استنتاج می­کرد. روش جدید بسیار کاراتر بود و خطای کمتری داشت اما نیازمند این بود که برای هر موضوع، مجموعه منتخب از جملات به دقت تعیین گردد. در سال 2007، مفهوم استخراج اطلاعات باز معرفی گردید. این سیستم ها توانایی گسترش به هر زمینه و هر موضوعی را دارند. ایده این روشها هم استفاده از اطلاعات نحوی موجود در متن و ایجاد ارتباط بین این اطلاعات و مفهوم نهفته در متن بود.

**ارزیابی سیستم های تشخیص موجودیت نامدار**

در این سیستمها، ورودی دنباله ای از کلمات و هدف شناسایی موجودیت نامدار است. برای اینکار باید شروع و پایان هر موجودیت و دسته مربوط به آن در متن تعیین گردد. به مثال زیر توجه کنید

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *رامین* | *مهمان* | *پرست،* | *سخنگوی* | *وزارت* | *امور* | *خارجه* | *دیروز* | *در* | *جمع* | *خبرنگاران* | *گفت* |
| PER | PER | PER | O | ORG | ORG | ORG | O | O | O | O | O |

به این مسئله می­توان به صورت یک مسئله دسته بندی کننده نگاه کرد که به هر کلمه ورودی یک برچسب می­زند. اما اینکار چندان درست نیست. زیرا در این مسئله ورودی دنباله ای از کلمات است و مثلا اگر یک کلمه با برچسب ORG داشته باشیم، احتمال اینکه کلمه بعدی نیز ORG باشد زیاد است. بنابراین روش ذکر شده همبستگی بین توکن ها در نظر نگرفته است.

استفاده از روش معمول دقت و فراخوان[[12]](#footnote-12) برای شناسایی موجودیت های نامدار چندان کارا نیست. به مثال زیر توجه کنید.

|  |  |  |  |
| --- | --- | --- | --- |
| *رامین* | *مهمان* | *پرست،* | *سخنگوی* |
| PER | PER | O | O |

در این مثال برچسب زننده[[13]](#footnote-13) ما بدنبال یافتن اسامی در متن است. به درستی دو توکن اول را برچسبPER زده است ولی توکن سوم را به اشتباه برچسب O زده است. در روش دقت و فراخوان چون کل عبارت «رامین مهمان پرست» یک نام است؛ اگر سیستم کل عبارات را درست تشخیص دهد، یکی به تعداد درست های تشخیص داده شده اضافه می­شود و در غیر این صورت یکی به تعداد غلط ها. حال در این شرایط علارغم اینکه سیستم جوابی نزدیک به درست داده و تنها بخشی از پاسخ را جا انداخته است؛ دو امتیاز منفی یکی برای اینکه پاسخ اش درست نبوده و دیگری برای اینکه یک نام را در متن نیافته است می­گیرد. این در حالیست که اگر سیستم کلا به همه توکن ها برچسب O میزد تنها یک خطا داشت. پر واضح است که این خلاف منطق است.

برای حل این مشکل پیشنهادات مختلفی برای ارزیابی سیستم های شناسایی موجودیت نامدار ارائه شده است. در این سیستم ها به برچسب زن، اعتبار جزئی برای حالاتی که موجودیت نامدار را به شکل تقریبا صحیح تشخیص داده است (مانند مثال بالا) می­دهند. یکی از این روشها امتیاز ماک[[14]](#footnote-14) است. البته چون محاسبه خود امتیاز ماک هم پیچیدگی های خاص خود را دارد؛ زیرا باید تعیین کنیم چگونه و به هر پاسخ نسبی چه امتیازی داده شود؛ بنابراین هنوز هم در بسیاری از منابع و مقالات از معیار اف یک[[15]](#footnote-15) برای گزارش نتایج سیستم خود استفاده می­کنند.

**مدل بیشینه عدم قطعیت و تخمین ممیزی[[16]](#footnote-16)**

تاکنون مدل های مولدی[[17]](#footnote-17) زیادی مانند دسته بندی کننده نایو بیز[[18]](#footnote-18) و مدل زبانی[[19]](#footnote-19) معرفی شده اند. ویژگی این مدل ها در این است که همگی توانایی تولید (سنتز) داده طبق مدل آموزش یافته را دارند. اما امروزه کاربردهای زیادی برای مدل های شرطی ممیزی[[20]](#footnote-20) در پردازش متن، صوت، استخراج اطلاعات و به طور کلی یادگیری ماشین دیده می­شود. ویژگی مشترک این مدل توانایی دسته بندی داده ها بر اساس مدل آموزش داده شده است. این رویکرد جدید به دلایل زیر صورت گرفته است

* کارایی و دقت بالا در دسته بندی دارند.
* می­توان به سادگی اطلاعات زبانی زیادی را به کمک این روش ها، مدل کرد.
* در تولید مدل های زبانی قابل استفاده مجدد کارآمدند.

در ابتدا تفاوت های مدل تولیدی و مدل تمایزی را بررسی میکنیم. اهمیت دانستن تفاوت های این دو مدل در انتخاب مدل مناسب برای حل مسئله است. دیده شده است که در بسیاری از موارد محققین به اشتباه برای حل یک مسئله دسته بندی از مدل های مولدی و به عکس برای حل مسئله سنتز داده از مدل های ممیزی استفاده کرده اند. البته نمیتوان به کلی انجام اینکار را رد کرد ولی بهتر است برای رسیدن به نتایج مناسب در انتخاب مدل دقت کرد. در هر دو دسته مدل، فرض میکنیم مجموعه داده به صورت داریم که در آنها d داده مشاهده شده و c کلاس داده که به صورت مخفی است. در مدل های مولدی (توزیع توام[[21]](#footnote-21))، احتمالات را روی جفت داده و کلاس مخفی به فرم قرار میدهند. (احتمال به صورت توزیع توام در نظر گرفته می­شود.) در واقع داده مشاهده شده را از روی اطلاعات مخفی تولید می­کنند. این روش ها تمام تکنیکهای کلاسیک پردازش متن شامل مدل چندگرمی[[22]](#footnote-22)، نایو بیز، مدل مخفی مارکوف[[23]](#footnote-23) و ... را در بر می­گیرند. هر جا تابع توزیع احتمال توام داده و کلاس را داشته باشیم، میتوانیم از این توزیع برای سنتز داده مشابه آنچه آموزش دیده ایم استفاده کنیم.

در نقطه مقابل، روشهای ممیزی (شرطی)، به شکل مستقیم تری دسته بندی که می­خواهیم انجام دهیم را مورد هدف قرار می­دهند. در واقع تابع احتمال را به صورت شرطی می­بینند. احتمال کلاس به شرط داده مشاهده شده . روش های ممیزی شامل رگرسیون لاجستیک[[24]](#footnote-24)، مدل بیشینه آنتروپی و میدان تصادفی شرطی می­شوند. البته بسیاری از دیگر روشهای یادگیری ماشین مانند ماشین بردار پشتیبان[[25]](#footnote-25) و شبکه عصبی نیز در دسته روشهای ممیزی برای دسته بندی قرار می­گیرند که در واقع به صورت مستقیم بر مبنای احتمالات قرار نگرفته اند.

یک روش برای نمایش تفاوت بین دو مدل، استفاده از مدلهای گرافی احتمال[[26]](#footnote-26) است. یک مدل گرافی، مدلی احتمالاتی است که در آن استقلال شرطی بین متغیرها توسط یک گراف نشان‌داده می‌شود. دو نوع مدل گرافی وجود دارد: مدل با گراف جهت‌دار که شبکه بیزی نامیده می‌شود و مدل با گراف ساده که میدان تصادفی مارکفی نامیده می‌شود (Bishop, 2006) (Daphne Koller, 2009). در مدل گرافی، متغییرها را با دایره و خطوط بین دایره ها نشاندهنده وابستگی مستقیم بین متغییرها است. برخی از متغییرها به صورت معمولی قابل مشاهده اند و برخی مخفی هستند. در شکل زیر، مدل گرافی احتمالی برای دو دسته بندی کننده رگرسیون لاجستیک (به نمایندگی از مدل ممیزی) و نایو بیز (به نمایندگی از مدل مولدی) نمایش داده شده است.



تصویر 6 مقایسه مدل گرافی احتمالی دو روش مولدی و ممیزی

در نایو بیز در زمان دسته بندی، کلمات مختلف متنی را که به ما داده شده است داریم (d1, d2,d3). می­خواهیم بر اساس آن کلاس داده (c) را تعیین کنیم. از دید احتمالاتی، ما احتمال اولیه کلاس و احتمال هر کلمه به شرط کلاس را داریم. جهت فلش از کلاس به داده (مولدی) است که نشان می­دهد که کلمات از کلاس تولید شده­اند. و در واقع چیزی را که واقعا مشاهده کرده ایم را پیش گویی نمی­کند. اما وضعیت در رگرسیون لاجستیک معکوس این حالت است. دوباره ما کلامات درون متن را مشاهده می­کنیم و به دنبال پیشگویی کلاس داده هستیم. اما این بار ما به شکل مستقیم یک احتمال را بر روی کلاس به شرط داده مشاهده شده قرار میدهیم .

بنابراین در مدلهای مولدی به دنبال توزیع توام احتمال داده و کلاس و بیشینه کردن درستنمایی احتمال هستیم. انجام اینکار همانطور که می­دانیم بدیهی است. انتخاب وزنها برای این مدل بر اساس شمارش وقوع رخدادهای مختلف یک چیز و سپس تقسیم کردن بر یک مخرج نرمال کننده انجام می­گیرد. این روش برای تخمین بسیار کاربردی است.

در نقطه مقابل، در مدل شرطی ممیزی، میخواهیم احتمال را محاسبه کنیم. در این مدل سعی می­کنیم به صورت مستقیم، احتمال درستنمایی شرطی را بیشینه کنیم. احتمال کلاس های مشاهده شده به شرط داده. در ادامه خواهیم دید که اینکار به مراتب مشکلتر بوده ولی در عوض مفیدتر است زیرا به شکل مستقیم به درستی و غلطی دسته بندی کننده مربوط شده و نتایج بهتری را تولید خواهد کرد.

برای بررسی بهتر این قضیه به مقاله ای در زمینه مقایسه مدل های مولدی و ممیزی در زمینه رفع ابهام معنای کلمه[[27]](#footnote-27) توجه کنید (Dan Klein, 2002). در این مقاله، هدف تعیین معنای صحیح کلمه از بین معانی مختلف آن در متن است. به عنوان مثال کلمه «شیر» در فارسی می­تواند معانی بسیار متفاوتی داشته باشد. شیر خوردنی، شیر سلطان جنگل، شیر آب و شیر به معنای عصاره هر چیزی. در این مقاله سعی می­شود با دیدن کلمه شیر درون متن (با کمک شواهد در اطراف کلمه) تشخیص دهیم کدامیک از این معانی مد نظر نویسنده بوده است. دو مدل که از هر نظر کاملا با هم یکسان اند برای حل مسئله در نظر گرفته شده است. هر دو فیچر های یکسان، روش هموارسازی یکسان[[28]](#footnote-28) و داده های آموزشی و آزمایشی یکسانی دارند. تنها تفاوت دو روش حل مسئله، در این بوده که در یکی از روش مولدی و دیگری ممیزی برای دسته بندی استفاده شده است. در جدول زیر نتایج حاصل از تحقیق را مشاهده می­کنید.



تصویر 7 مقایسه مدل ممیزی و مولدی روی نتایج حاصل از آزمایش

اگر به نتایج حاصل روی داده های آزمایشی توجه کنید می­بینید که روش ممیزی کارایی بالاتری نسبت به مدل مولدی دارد (حدود 2.5 درصد). در این مثال روش مولدی مورد استفاده نایو بیز است. نکته دیگر این است که با توجه به نتایج داده های آموزشی متوجه می­شویم که روش تمایزی حدود 12 درصد کارایی بالاتری دارد. ممکن است در نظر اول این نتیجه ما رو خوشحال کند اما در واقع نشان می­دهد که روش ممیزی توانایی بالایی برای بخاطر سپردن داده های آموزشی دارد و بسیار مستعد بیش برازش[[29]](#footnote-29) است.

**استخراج ویژگی از متن در روش های ممیزی**

در این بخش به بررسی روشهای استخراج ویژگی از متن برای استفاده در دسته بندی کننده های ممیزی می­پردازیم. ویژگی[[30]](#footnote-30) به معنای گواهی کوچکی است که ارتباطی بین آنچه که مشاهده کردیم (همان داده D) و یک کلاس داده C برقرار می­کند. ویژگی، یک تابع با مقدار محدود و حقیقی است. در ادامه چند نمونه از ویژگی های مورد استفاده در روش های ممیزی برای شناسایی موجودیت نامدار را مشاهده می­کنید.

|  |
| --- |
|  |

و داده آموزشی زیر را داریم.

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 3 | 2 | 1 |
| PERSON | DRUG | LOCATION | LOCATION |
| saw Sue | taking Zantac | in Québec | in Arcadia |

ویژگی یک (بنفش) سه شرط دارد. اولا کلاس داده LOCATION باشد، دوما کلمه قبلی «in» باشد و خود کلمه با حرف بزرگ شروع شده باشد. هر سه این ویژگی ها در مورد داده های آموزشی دو ستون سمت چپ (1و2) صادق هستند. ولی در مورد دو ستون سمت راست (3و4) درست نیستند. ویژگی دوم می­گوید کلاس داده LOCATION است و کلمه، درون خود کاراکترهای لاتین دارد. این ویژگی تنها در مورد ستون 2 درست است. و سرانجام ویژگی سوم می­گوید، کلاس داده DRUG است و حرف انتهایی «c» باشد. این ویژگی تنها ستون 3 را انتخاب می­کند. در هنگام بررسی ویژگی همیشه فرض می­کنیم که یک محل مشخص در متن داریم که در آن نقطه در حال بررسی هستیم. مثلا در زمان بررسی ستون سوم فرض کردیم که کلمه در حال بررسی «Zantac» است (w=Zantac) و کلمه قبل نیز «taking» است (w-1=taking).

در این مرحله، مدل ممیزی به هر ویژگی یک وزن اختصاص می­دهد. وزن نیز یک عدد حقیقی است. مقدار مثبت برای یک ویژگی نشان می­دهد که آن ویژگی در مورد داده آموزشی صادق بوده است و چیزی است که در متن واقعی نیز رخ داده است. به عنوان نمونه ویژگی اول چون در مورد دو ستون 1و2 صادق است، ممکن است وزنی برابر با 0.9 بگیرد. در نقطه مقابل مقدار منفی برای یک ویژگی، نشان می­دهد که احتمال وقوع آن ویژگی پایین است و آن ویژگی در داده آموزشی صحیح نبوده است. به عنوان نمونه به ویژگی زیر توجه کنید.

این ویژگی به داده ستون اول کلاس DRUG را نسبت می­دهد که غلط است و ما میخواهیم نشان دهیم که احتمالا این ویژگی غلط است و به آن وزن -0.6 را نسبت می­دهیم.

برای محاسبه وزن ویژگی ها، از دو مقدار شمارش تجربی و مدل امید ریاضی[[31]](#footnote-31) ویژگی استفاده می­کنیم. که با دو معادله زیر تعریف می­شوند.

در معدله اول که مربوط به شمارش تجربی است، به سادگی تنها روی داده های آموزشی حرکت کرده و تعداد دفعاتی که ویژگی مورد نظر در آنها درست است را می­شماریم. در روش دوم، توزیع احتمال بر روی جفت داده و کلاس را داریم. با استفاده از آن احتمال تمام جفت کلاس و داده ها را در نظر گرفته و مقدار امید ریاضی ویژگی را محاسبه می­کنیم.

به شکل خاص در کاربرد پردازش متن، ویژگی های تعریف شده دارای دو خاصیت زیر هستند:

1. یک تابع شاخص با مقادیر بله یا نه بر روی خواص داده ورودی.
2. یک کلاس داده

یک ویژگی دلخواه در پردازش متن به صورت زیر نمایش داده می­شود.

یک ویژگی دلخواه بر روی کلاس c و داده d با یک مسند تطبیقی[[32]](#footnote-32) با نام و یک کلاس نمایش داده می­شود. مسند تطبیقی مانند اینکه انتهای کلمه حرف «c» دارد یا کلمه قبلی «in» است یا نه؟ و کلاس داده مانند PERSON یا LOCATION است. مقدار خروجی ویژگی به شکل کلی می­تواند یک عدد حقیقی باشد ولی اینجا ویژگی های ما مقدار یک را در صورت تطبیق با داده و کلاس و در غیر این صورت مقدار صفر را بازمی­گردانند. لذا می­توان گفت هر ویژگی، زیر مجموعه ای از داده را تعیین کرده و یک برچسب برای آن پیشنهاد می­دهد. در مدلهای مبتنی بر ویژگی، تصمیمی که برای یک داده در یک نقطه از متن اتخاذ می­گردد، تنها بر مبنای ویژگی هایی است که در آن نقطه فعال (مقدار یک) هستند.

در ادامه به برخی از موارد استفاده از ویژگی های تعریف شده در پردازش متن می­پردازیم. مقاله(Tong Zhang, 2001) یک نمونه معروف در زمینه طبقه بندی متن[[33]](#footnote-33) است. ویژگی هایی که در این مقاله مورد استفاده قرار گرفته اند، کیسه لغات[[34]](#footnote-34) است. به طور دقیق ویژگی ها برابر با حضور لغت در متن و کلاس متن است. البته از تمام کلمات موجود در متن به عنوان ویژگی استفاده نمی­شود. یک مرحله انتخاب ویژگی های مهم وجود دارد تا ویژگی های قابل اطمینان تعیین گردند. در این مرحله کلمات نادر حذف می­گردند. همچنین کلمات بسیار پرکاربرد نیز حذف می­شوند زیرا به نظر می­رسد که مفهوم چندانی را در خود نداشته باشند. دادگان مورد استفاده در این مقاله، مجموعه اخبار خبرگزاری رویترز است. معیار اف یک بدست آمده برای روش نایو بیز برابر با 77.0 درصد است. سپس آنها چندین روش ممیزی را هم آزمایش کرده­اند. روش رگرسیون خطی دقتی برابر با 86.0 درصد، رگرسیون لاجستیک 86.4 درصد و ماشین بردار پشتیبان 86.5 درصد را بدست آوردند. کارایی روش های ممیزی تقریبا با هم برابر در یک سطح بوده است. اما تفاوت بزرگ در این نتایج بین نایو بیز که یک مدل مولدی است با سایر روشها است. تمام مدل های ممیزی نتایج بسیار بهتری را نسبت به نایو بیز بدست آورده اند. نکته مهم دیگر در مقایسه نتایج روش های ممیزی این است که نوع تکنیک بکار رفته تاثیر چندانی در بالا بردن کارایی نداشته است. چیزی که در اینجا مهم است، نحوه تعریف ویژگی ها و کیفیت آنها است. پس بهتر است برای بالا بردن کارایی روی تعیین ویژگی های حاوی اطلاعات مفید کار شود تا تغییر دسته بندی کننده. نکته دیگری که مقاله به آن اشاره می­کند در اهمیت استفاده از روش های هموارسازی هنگام بکارگیری مدل ممیزی است.

دسته بندی کننده بیشینه آنتروپی موارد کاربرد بسیاری دارد. در هر مسئله ای که مجموعه ای از داده ها را داریم و می­خواهیم به آنها یک کلاس را نسبت دهیم، امکان استفاده از این دسته بندی کننده وجود دارد. در ادامه به چند مورد از کاربرد های آن در مقالات اشاره می­کنیم. مقاله (Mikheev, 2000) به تعیین محدوده شروع و پایان جملات پرداخته است. در این مقاله برای هر نقطه (.) موجود در متن تشخیص می­دهد که آیا انتهای یک جمله است یا نه؟ مقاله (Pang, 2002) در زمینه تشخیص نظر کاربران است. در این مقاله نیز از دسته بندی کننده ممیزی استفاده شده است. ویژگی های مورد استفاده شامل کلمات، جفت کلمات (مدل دو-گرمی)، برچسب مقوله نحوی و ... است. مقاله (Ratnaparkhi, 1998) به استخراج گروه قیدی[[35]](#footnote-35) از داخل متن پرداخته است.

**استفاده از ویژگی های استخراج شده در یک دسته بندی کننده خطی**

در این بخش به معرفی دسته بندی کننده های خطی و کاربرد ویژگی های استخراج شده در آن می­پردازیم. این دسته بندی کننده ها، خطی هستند زیرا در انتهای الگوریتم؛ مجموعه ای از ویژگی ها را خواهیم داشت که بر روی آنها یک تابع خطی را اعمال می­کنیم. و از روی نتایج حاصل از این تابع خطی امتیازی به هر کلاس داده می­شود. این کار با اختصاص دادن یک وزن به هر ویژگی انجام می­گیرد. در مرحله بعد برای هر داده مشاهده شده، تمام کلاس های ممکن را در نظر میگیریم و مشاهده می­کنیم به ازای هر کلاس کدامیک از ویژگی ها برقرار است. امتیاز هر کلاس با مجموع حاصلضرب وزن در ویژگی های فعال برای آن محاسبه می­شود. در مثال زیر اگر بخواهیم تعیین کنیم کلمه Québec کدامیک از برچسب LOCATION، PERSON یا DRUG را دارد از رابطه زیر استفاده میکنیم. (مثال in Québec)

بر اساس این رابطه، کلاس C ای که مقدار بالا را ماکزیمم کند انتخاب میگردد. و چنانچه وزن ها و ویژگی ها به خوبی تعیین شده باشند، امیدوار خواهیم بود که برچسب LOCATION برای کلمه Québec انتخاب گردد. (فرض کنید برای کلاس PERSON هیچ ویژگی در این مثال درست نباشد، برای کلاس LOCATION دو ویژگی با وزن های 1.8 و -0.6 و برای کلاس DRUG یک ویژگی با وزن 0.3 برقرار باشد.)

روشهای زیادی برای تعیین وزن ویژگی­ها در دسته بندی کننده وجود دارد.

* پرسپترون: یک نمونه با دسته بندی غلط را یافته و تلاش میکند با تغییر وزن های اولیه دسته بندی آن را اصلاح نماید.
* روش­های مبتنی بر حاشیه مانند ماشین بردار پشتیبان
* روش­های بهینه سازی مختلف مانند نزول در جهت گرادیان[[36]](#footnote-36) یا ال­بی‌اف‌جی‌اس[[37]](#footnote-37).

در این بخش دسته بندی کننده ماکزیمم آنتروپی یا به اختصار (Maxent) را مورد بررسی قرار می­دهیم که یک دسته بندی کننده نمایی یا لگاریتم-خطی[[38]](#footnote-38) است. البته این گروه دسته بندی کننده­ها نام­های دیگری مانند لجستیک یا مدل گیبس[[39]](#footnote-39) را هم دارند. ایده اصلی این گروه در تولید یک مدل احتمالاتی از روی ترکیب خطی است. از آنجاییکه مقادیر وزن ها می­تواند منفی یا مثبت باشد، این مجموع نیز ممکن است مقادیر مثبت یا منفی بگیرد. می­دانیم مقدار منفی برای احتمال غیر قابل قبول است. پس باید راهی بیابیم که این مجموع را همواره مثبت کنیم. برای این منظور همواره از این مجموع Exp میگیریم. در واقع عدد نپر را به توان این مجموع میرسانیم . این مقدار همیشه مثبت است اما لزوما یک احتمال نیست زیرا ممکن است مقدار بیشتر از یک هم بگیرد. برای حل این مشکل نیز نتایج را نرمال می­کنیم. بنابراین احتمال هر کلاس برای یک داده و مقادیر وزن مشخص از رابطه زیر محاسبه میگردد:

با در نظر گرفتن مثال قبلی و ویژگی های تعریف شده، احتمال برچسب های مختلف برای کلمه Québec از روابط زیر محاسبه می­گردد.

|  |  |  |
| --- | --- | --- |
|  |  |  |

به خاطر استفاده از تابع نمایی، تمامی احتمال ها مثبت و به خاطر استفاده از ترم نرمال کننده، حاصل جمع سه احتمال برابر با یک شده است.

با داشتن این مدل، وزن ها () را به گونه ای تعیین می­کنیم که تابع درست نمایی شرطی داده بر اساس مدل ماکزیمم گردد. بنابراین نه تنها دسته بندی کننده را تولید می­کنیم، بلکه تابع توزیع احتمال بر روی آنها را نیز می­سازیم.

**استفاده از مدل توالی[[40]](#footnote-40) برای حل مسائل برچسب گذاری در استخراج اطلاعات**

همانگونه که قبلا اشاره شد، برای حل یک مسئله برچسب گذاری در استخراج اطلاعات میتوان به آن بصورت یک مسئله دسته بندی کلاسیک نگاه کرد. تک تک توکن های متن را استخراج نمود. آن ها را به صورت مستقل به دسته بندی کننده داد و برچسب آن را تعیین نمود. اما مشکل این روش در نظر نگرفتن همبستگی موجود در متن و ارتباط بین توکن های متوالی درون جمله است. با اینکار ویژگی توالی ذاتی کلمات در نظر گرفته نمیشود. به عنوان مثال در زبان انگلیسی تقریبا غیر ممکن است که یک فعل بعد از یک حرف تعریف در جمله بیاید. حال اگر در حال تعیین برچسب مقوله نحوی برای کلمات درون یک جمله انگلیسی باشیم، چنانچه از روش مطرح شده استفاده کنیم، چه بسا در مواردی به دو توکن متوالی برچسب ]حرف تعریف – فعل[ بزنیم که مطابق آنچه مطرح شد به احتمال زیاد اشتباه است. با در نظر گرفتن این مدل اطلاعات زیادی که در توالی توکن ها است از دست میرود. لذا به منظور بالا بردن دقت برچسب زننده خود، باید به نحوی اطلاعات توکن های مجاور را در تصمیم گیری خود منظور کنیم.

در ادامه بحث توضیحات را روی مسئله شناسایی موجودیت نامدار که یک مسئله کلاسیک در زمینه برچسب گذاری توالی در استخراج اطلاعات است، متمرکز میکنم. برای استفاده از مدل توالی در شناسایی موجودیت نامدار نیازمند داده های آموزشی با ناظر هستیم. یعنی مجموعه از متون آموزشی داریم که در آنها کلمات بر اساس موجودیت شان برچسب صحیح خورده اند. بدین ترتیب مراحل زیر را برای آموزش سیستم داریم:

* یک مجموعه متن که نماینده متون آزمایشی ما هستند و در آنها موجودیت هایی که به دنبال یادگیری آنها هستیم وجود دارد.
* در این مرحله بر روی کلمات متون حرکت می­کنیم. برچسب موجودیت صحیح را به آن می­دهیم و اگر شامل هیچکدام از موجودیت های ما نمی­شد، برچسب O که نماینده Other است را به آن میدهیم.
* در بخش یادگیری ماشین، مدلی برای استخراج ویژگی که مناسب برای متن و برچسب های هدف باشد طراحی میکنیم.
* آموزش یک دسته بندی کننده توالی برای تعیین برچسب های هدف در متن.

در این مرحله مدل لازم برای برچسب زنی داده های آموزشی تولید شده است. به منظور آزمایش مدل مراحل زیر را دنبال میکنیم:

* جمع آوری مجموعه ای از متون آزمایشی
* اعمال مدل یادگیری شده برای استنتاج برچسب مناسب برای کلمات درون متن. این مدل به ما احتمال هر برچسب را به ازای کلمه ورودی میدهد.
* در خروجی برچسب مناسب را بر اساس احتمالات محاسبه شده تولید می­کنیم.

برای فهم بهتر مطلب یک مثال می­زنیم. در مثال زیر دو روش برای برچسب گذاری مشاهده می­شود.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| painting | new | ‘s | Picasso | Pablo | Sue | showed | Fred |  |
| O | O | O | PER | PER | PER | O | PER | IO Labeling |
| O | O | O | I-PER | B-PER | B-PER | O | B-PER | IOB Labeling |

در روش اول (IO Labeling) به هر کلمه در صورتیکه موجودیت داشته باشد برچسب آن را میدهیم و در غیر این صورت برچسب O یعنی بدون موجودیت میدهیم. اما مشکلی در این برچسب گذاری وجود دارد! در جمله بالا سه نام Fred، Sue و Pablo Picasso وجود دارد. اما برچسب گذاری ما تنها دو نام را نشان میدهد. در واقع هیچ روشی برای اینکه نشان دهیم در یک دنباله برچسب، نام دو نفر وجود دارد نداریم. برای حل این مشکل می­توان از روش دوم استفاده کرد.

در روش دوم (IOB Labeling) در صورتیکه یک موجودیت در دنباله شروع شود به آن برچسب B-Entity میدهیم. مثلا به شروع نام یک فرد مانند Pablo برچسب B-PER میدهیم (B ابتدای کلمه Begin است). و در صورتیکه کلمه ادامه موجودیت قبلی باشد به آن برچسب I-Entity میزنیم. مثلا به کلمه Piccaso چون ادامه نام است برچسب I-PER میزنیم (I ابتدای کلمه Inside است). مشابه روش قبل در صورتیکه کلمه موجودیت نباشد به آن برچشب O میدهیم. با این روش مشکل روش اول حل میشود. اما برچسب گذاری دوم از نظر هزینه محاسباتی مشکلاتی دارد. در روش اول اگر فرض کنیم c موجودیت داریم. در مجموع برچسب خواهیم داشت (به اضافه یک برای برچشب O است). اما در روش دوم برچسب خواهیم داشت (ضریب 2 برای شروع و پایان ها است). در نگاه اول این تفاوت ناچیز به نظر می­رسد. اما توجه کنید که اگر از مدل توالی برای برچسب زنی استفاده کنیم باید حداقل مجموعه حالات را به توان مرتبه مدل توالی برسانیم یعنی حداقل به توان 2. بنابراین برای روش اول حالت و برای روش دوم خواهیم داشت. بنابراین در نهایت استفاده از روش دوم موجب پایین آمدن شدید سرعت اجرا میگردد. با این حال که روش دوم بر اساس برخی نقطه نظرات روش درست برای برچسب زنی است؛ بسیاری از مقالات نیز از همین روش استفاده میکنند؛ اما به سه دلیل ما روش اول را برای برچسب زنی برمی­گزینیم.

1. بسیار سریع تر است. آموزش آن با داده کمتر و در زمان کوتاه تری امکان پذیر است.
2. مشکل فقط زمانی رخ میدهد که ما دو موجودیت با کلاس یکسان را به صورت متوالی در ورودی داریم. مثلا نام دو نفر پشت هم آمده است. این حالت در زبان انگلیسی به ندرت رخ می­دهد. در زبان فارسی هم که هدف اصلی ما است، این حالات به خاطر وجود علامات و کلمات ربط مانند «و» و «،» یا وجود کلمه «را» بعد از مفعول، کمتر از زبان انگلیسی رخ می­دهد.
3. سیستم هایی که از روش دوم استفاده میکنند، در عمل به ندرت می­توانند در خروجی پایان یک موجودیت و شروع همان نوع موجودیت را بدرستی تشخیص دهند و عموما برای جمله مورد مثال خروجی زیر را تولید می­کنند.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| painting | new | ‘s | Picasso | Pablo | Sue | showed | Fred |  |
| O | O | O | I-PER | I-PER | B-PER | O | B-PER | IOB Labeling in real practice |

که این نوع دسته بندی دقیقا معادل روش اول است و مزیتی نسبت به آن ندارد. لذا در عمل استفاده از روش اول علارغم اندکی مشکل منطقی تر است.

**ویژگی های مورد استفاده در روش مدل توالی**

اولین ویژگی واضح خود کلمات هستند. برای هر کلمه یک ویژگی برای بررسی احتمال قرار گرفتن آن کلمه در کلاس مورد نظر داریم. لذا این ویژگی در واقع مانند یک لغت نامه یادگیری شده از کلمات برای هر کلاس عمل می­کند. همچنین ویژگی هایی برای کلمات قبل و بعد از کلمه در حال پردازش قرار می­دهیم. مثلا میدانیم که بعد از کلماتی مانند «در» یا «به» احتمال آمدن یک موجودیت مکان با برچسب LOC بیشتر است. (علی به مدرسه رفت. حسن در خیابان انقلاب ماند.) چنانچه نتایج پردازش های زبانی دیگر مانند برچسب مقوله نحوی را داشته باشیم، بهتر است از آنها هم به عنوان ویژگی استفاده کنیم. برچسب مقوله نحوی کلمه در حال پردازش، کلمه قبل و بعد از آن مورد استفاده قرار می­گیرند.

تمام این ویژگیها تنها به داده ورودی توجه می­کنند و می­توان با آنها یک دسته بندی کننده معمولی بدون مدل توالی ساخت. تنها زمانی دسته بندی کننده ما در ساختار مدل توالی قرار می­گیرد که اطلاعات برچسب های کلمات مجاور را نیز در تعیین برچسب کلمه جاری مورد استفاده قرار دهد. به عنوان مثال وقتی در حال پردازش جمله «نام او شهید ابراهیم همت بود.» هستیم. اگر دسته بندی کننده توالی ما متوجه شود که توکن «ابراهیم» برچسب نام فرد (PER) دارد، بنابراین با احتمال زیاد برچسب توکن بعدی «همت» نیز PER خواهد بود. زیرا نام افراد معمولا بیش از یک توکن درازا دارد. و به این صورت میتوان ویژگی هایی داشت که این یک دنباله را مدل کنند.

یک دسته دیگر از ویژگی ها که در شناسایی موجودیت نامدار مورد استفاده اند؛ ویژگی حروف سازنده زیر رشته[[41]](#footnote-41) است. این ویژگی ها برای دسته بندی کلمات میتوانند بسیار کارآمد باشند. روش کار به این صورت است که یک زیر رشته از حروف را به عنوان ویژگی در نظر می­گیریم. مثلا اگر میخواهیم اسامی داروهای به زبان انگلیسی را استخراج کنیم، میتوان زیر رشته «oxa» را به عنوان ویژگی در نظر گرفت. در مثال زیر چند ویژگی حروف سازنده زیر رشته برای کاربرد تشخیص 5 دسته موجودیت شامل اسامی داروها، شرکت ها، فیلم ها، مکانها و اشخاص را مشاهده می­کنید.



تصویر 8 بررسی ویژگی های مختلف روی متن برای تشخیص موجودیت نامدار

همانطور که در شکل میبینید، ویژگی زیر رشته ای «oxa» با دقت 100 درصد فقط در اسامی داروها در داده آموزشی دیده شده است. و برای دسته بندی کردن بسیار مناسب است. از طرف دیگر ویژگی زیر رشته ای «:» اغلب در اسامی فیلمها به زبان انگلیسی دیده شده است. زیر رشته «field» نیز در اسامی اشخاص، فیلم ها و مکان ها بیشتر دیده شده است.

ویژگی دیگری که کاملا مکمل ویژگی کاراکترهای سازنده زیر رشته است، دنباله شکل کلمات[[42]](#footnote-42) است. ایده این روش اولین بار توسط مایکل کالینیز [?] مطرح شد. در این روش کلمات را بر اساس شکل ظاهریشان به یکسری کلاس نگاشت میکنیم. این کلاس ها نمایشی ساده از کلمات اند که خواصی از آنها مانند طول کلمه، بزرگ و کوچک بودن آن (برای انگلیسی)، استفاده از اعداد یا علائم نگارشی درون آن را در خود دارند. شکل زیر نمونه ای از این کلاس ها را برای زبان انگلیسی ارائه می­کند.



در این سیستم استخراج ویژگی؛ هر حرف بزرگ به X، هر حرف کوچک به x، هر عدد به d و علائم نگارشی مانند -:, به خودشان نگاشت می­شوند. علاوه بر این در تولید کلاس ها نکته دیگری نیز مد نظر قرار گرفته است. ایده این است که از هر کلمه تنها دو حرف شروع و پایانی آن مورد استفاده قرار بگیرد. بنابراین اگر کلمه کمتر یا مساوی 4 حرف داشته باشد؛ کل آن در کلاس نماینده اش نگاشت میشود، در غیر این صورت برای تمام حروف بین دو حرف اول و آخر کلمه، فقط نوع حروف را در نمایش کلاسی میآوریم. در مثال بالا برای محاسبه کلاس کلمه «Varicella-zoster» به ازای Va ابتدای کلمه Xx، به ازای er انتهای کلمه xx و به ازای «ricella-zost» تنها x- که نشاندهنده نوع حروف در این زیر رشته است در کلاس نماینده شان آمده است که مجموعا موجب میشود تا این کلمه به کلاس Xx-xxx نگاشت شود. روشهای زیاد دیگری نیز وجود دارند که میتوان این کلاس ها را تعریف کرد. ولی نکته اصلی این است که این کلاسها اطلاعات کلی از ظاهر و حروف سازنده کلمه را در خود نهفته اند. بدین ترتیب کلمات را فضای بسیار کوچکتر و چگالتری میتوان نگاشت کرد.

**مدل توالی بیشینه عدم قطعیت[[43]](#footnote-43)**

داده ها در اغلب مسائل پردازش متن به صورت توالی از حروف، کلمات، عبارات، خطوط و جملات هستند و کار ما در بسیاری از موارد برچسب زدن به این توالی رشته ای است. یک نمونه بارز از این مسئله، برچسب مقوله نحوی است. نمونه های دیگر شامل شناسایی موجودیت نامدار، برچسب نقش معنایی[[44]](#footnote-44)، شناسایی انتهای جملات، قطعه بندی[[45]](#footnote-45) و تشخیص انتهای کلمه در زبان چینی است. در مدل توالی بیشینه عدم قطعیت، که به آن مدل مارکوف بیشینه عدم قطعیت[[46]](#footnote-46) نیز گویند، دسته بندی کننده به ازای هر توکن ورودی یک تصمیم میگیرد. این تصمیم مشروط بر شواهد حاصل از مشاهدات و تصمیمات قبلی است. مثال زیر کاربرد این روش در برچسب گذاری مقوله نحوی را نمایش می­دهد. برچسب سه توکن اول تعیین شده اند.



تصویر 9 استخراج ویژگی در مدل توالی بیشینه آنتروپی

در نقطه تصمیم گیری ویژگی های مورد استفاده برای دسته بندی کننده شامل کلمه جاری، کلمه بعدی، کلمه قبلی، برچسب کلمه قبلی، برچسب دو کلمه قبلی به صورت متصل مورد استفاده است. همچنین سایر ویژگی های مشابه آنچه قبلا بررسی شد نیز کاربرد دارند. ایده به کار رفته در دسته بندی نیز مشابه دسته بندی کننده ماکزیمم عدم قطعیت است که پیشتر مورد بررسی قرار گرفت. شکل زیر کلیات نحوه حرکت از دسته بندی کننده ابتدایی بیشینه عدم قطعیت به مدل توالی بیشینه عدم قطعیت را نشان می­دهد.



تصویر 10 کلیات نحوه حرکت از دسته بندی کننده ابتدایی بیشینه عدم قطعیت به مدل توالی بیشینه عدم قطعیت

در این شکل داده های توالی در بالا سمت چپ قرار دارند. هدف دسته بندی در مرحله توالی است. همانطور که در بالا راست دیده میشود، کلمات را در ورودی داریم و میخواهیم به هر کدام برچسب مناسب بزنیم. برای اینکار ما به هر تصیمیم به صورت مستقل نگاه میکنیم. به ازای هر تصمیم مجموعه از داده ها به صورت محلی (پایین سمت چپ) وجود دارد. این داده ها شامل کلمه جاری، کلمه قبلی، برچسب کلمه قبلی و ... است. سپس بر روی آن داده ها به استخراج ویژگی می­پردازیم. حال ما ویژگی های محلی را داریم، برچسب ها را هم بخاطر استفاده از یادگیری با ناظر داریم. با استفاده از آنها یک مدل بیشینه عدم قطعیت می­سازیم. در این مرحله تمام مسائلی که قبلا مطرح شده بود شامل بهینه سازی و هموارسازی های لازم انجام می­گیرد. در انتها یک دسته بندی کننده داریم.

چنانچه در هر مرحله برچسب توکن تعیین گردد و به مرحله بعد برویم، در واقع از یک روش حریصانه[[47]](#footnote-47) برای تعیین برچسب ها استفاده کرده ایم. اما در بسیاری از موارد بدنبال بهترین دنباله برچسب هستیم و علاقه مندیم تا فضای حالات برچسب ها را بهتر جستجو کنیم. زیرا مثلا فرض کنید به توکن اول برچسب A را داده ایم. بعد یک توکن دیگر را پردازش کرده ایم و به توکن سوم رسیدیم. در این مرحله بر اساس برخی شواهد متوجه میشویم که برچسبی که به توکن اول داده ایم نادرست بوده و به تبع آن برچسب توکن دوم نیز غلط است. حال ناگزیریم دوباره به توکن اول بازگشته، تصمیم خود را اصلاح کنیم و کار را از ابتدا تکرار کنیم. برای حل این مشکل چند روش وجود دارد. در ادامه با دو روش آشنا می­شویم.

**استنتاج پرتویی[[48]](#footnote-48)**

اولین این روش های استنتاج پرتویی است. در این روش به جای اینکه تنها بهترین جواب را نگهداریم، k تا از بهترین دنباله نتایج را ذخیره میکنیم. هر توالی از پاسخ ها را جداگانه ادامه میدهیم. بنابراین برای هر توکن ورودی باید k بار استنتاج انجام دهیم. این روش بسیار سریع بوده و پیاده سازی آن نیز آسان است. عموما اندازه k برابر با 3 یا 5 است. این اندازه برای استنتاج پرتویی نشان داده است که در بیشتر موارد میتواند بهترین توالی برچسب ها را محاسبه کند. ولی در بسیاری از موارد هم ممکن است که بهترین دنباله از برچسب ها از مجموعه پاسخ های استنتاج پرتویی خارج انداخته شود و در نتیجه جواب حاصل بهترین پاسخ توالی بر چسب نباشد. لذا در این روش هیچ تضمینی برای تولید بهترین پاسخ وجود ندارد.

**استنتاج ویتربی[[49]](#footnote-49)**

این روش بهتر از استنتاج پرتویی عمل میکند. در این روش بهترین توالی برچسب ها محاسبه می­گردد. به آن در پردازش متن استنتاج ویتربی یا الگوریتم ویتربی گفته می­شود. زیرا آقای اندرو ویتربی الگوریتم های زیادی برای حل این مشکل ارائه کرده است. پایه این روش بر برنامه نویسی پویا[[50]](#footnote-50) است. در آن یک پنجره کوچک تاثیر گذاری در نظر می­گیریم. برای انجام هر تصمیم گیری، پارامترهای تاثیر گذار تنها درون پنجره قرار دارند و هرچیزی خارج از آن را در نظر نمیگیریم. در این شرایط می­توان با استفاده از الگوریتم های برنامه نویسی پویا بهترین توالی برچسب را محاسبه نمود. مزیت آن در تضمین ارائه بهترین پاسخ است. مشکلات آن هم در سختی پیاده سازی و مجبور کردن کاربر به استفاده از برچسب های توکن های قبلی فقط در محدوده پنجره تعریف شده است.

**میدان تصادفی شرطی[[51]](#footnote-51)**

میدان تصادفی شرطی یکی دیگر از روشهای برچسب گذاری احتمالاتی توالی است. با رابطه زیر ارائه میگردد.

احتمالات c و d روی کل دنباله تعریف می­شوند و مربوط به یک نقطه خاص در فضای مسئله نیستند. در نظر اول اینکار بسیار مشکل به نظر میرسد زیرا فضای حالات c و d نمایی است. اما به شرط حفظ ویژگی های به صورت محلی می­توان جوری برنامه نویسی پویا را انجام داد که تابع درست نمایی شرطی[[52]](#footnote-52) دنباله را به درستی محاسبه نمود. آموزش مطمئنا نسبت به سایر روشهای استنتاج کند تر است ولی از نظر تئوری برتری هایی نسبت به مدل مارکوف بیشینه عدم قطعیت دارد. این روش در بسیار از مقالات به عنوان لبه تکنولوژی[[53]](#footnote-53) برای برچسب گذاری توالی مورد استفاده است. اما لازم به ذکر است که در عمل، نتایج حاصل از این روش نزدیک به نتایج مدل مارکوف بیشینه عدم قطعیت است. در چنین حالاتی چیزیکه تاثیر بیشتری دارد، انتخاب ویژگی های قدرتمند و مناسب است. و استفاده از مدل مارکوف بیشینه عدم قطعیت نیز برای برچسب گذاری توالی کاملا منطقی و قابل قبول است.

**ارتباط استخراج اطلاعات آزاد[[54]](#footnote-54) با برچسب نقش معنایی:**

هدف در برچسب زنی نقش معنایی، تحلیل افعال و اسامی در متن برای تعیین آرگومانهای آنها و نگاشت آنها به یک قاب معنایی برای تعیین نقش هر آرگومان در آن قاب است (Schmitz, 2012). یک مزیت برچسب نقش معنایی توانایی آن در تولید روابط چندگانه در جمله است. در جمله «علی با سنگ شیشه را شکست» در صورت داشتن منابع زبانی-معنایی مناسب مانند شبکه واژگان[[55]](#footnote-55) یا شبکه قاب[[56]](#footnote-56) می­توان رابطه 4 تایی زیر را استخراج کرد. شکستن(Agent؟علی، Patient؟شیشه، Instrument؟سنگ) اما مشکل برچسب نقش معنایی در محدود شدن به منابع زبانی است. این سیستمها عموما بدون داشتن این منابع نمی­توانند کار کنند. در نقطه مقابل استخراج اطلاعات آزاد توانایی کار روی هر موضوع در هر زمنیه ای بدون نیاز به منبع زبانی خاصی را داد (Schmitz, 2012). البته مطالعاتی اخیر (Janara Christensen, 2011) نشان داده است که با انجام برخی پس پردازش ها می­توان برچسب نقش معنایی را به استخراج اطلاعات آزاد تبدیل کرد.

**پیشینه تحقیق در زمینه استخراج رابطه[[57]](#footnote-57)**

همانگونه که استخراج اطلاعات به دو دسته استخراج اطلاعات آزاد و غیر آزاد (لفظ آزاد اشاره به استخراج هرگونه اطلاعات بدون هدف از پیش تعیین شده و محدوده مشخص از روابط از متن و استخراج اطلاعات غیر آزاد اشاره به استخراج اطلاعات با هدف معین و از روی مجموعه مشخص از روابط در متن را دارد.) تقسیم میگردد؛ استخراج رابطه هم بسته به نوع سیستم اصلی به دو دسته آزاد و غیر آزاد تقسیم بندی میگردد. البته معمولا به جای استفاده از لفظ استخراج اطلاعات غیر آزاد از همان لفظ استخراج اطلاعات استفاده میکنند. پس در واقع در تمام مقالاتی که اشاره به استخراج اطلاعات شده است، منظور استخراج اطلاعات غیر آزاد است. در این مقالات یک تعداد مشخص از روابط از پیش تعریف شده که معمولا از یک پایگاه داده سلسله مراتبی حقایق مانند (Freebase, 2007)بدست آمده اند، داریم. مثلا رابطه /people/person/place\_of\_birth را داریم. هدف بررسی تمام جفت موجودیت[[58]](#footnote-58) های ذکر شده در متن و تعیین نوع رابطه بین آنها است. اگر در مجموع n تا رابطه در پایگاه داده سلسله مراتبی ما وجود داشته باشند، این روشها n+1 (یک برچسب هم هیچ کدام از این روابط -None- است) برچسب را به جفت موجودیت ها میدهند. در نقطه مقابل، روشهای آزاد استخراج رابطه قرار دارند. این روشها هیچ گونه اطلاعات قبلی مانند یک پایگاه داده سلسله مراتبی یا وضعیت قرار گرفتن موجودیت های متن را ندارد. بلکه با توجه به ساختار لغوی و نحوی زبان در گروه های اسمی و فعلی متن به شناسایی روابط میپردازند. این روشها دارای فراخوان بسیار بهتری نسبت به روشهای غیر آزاد هستند. اما مشکل آنها این است که اطلاعات یافته شده توسط آنها از ارزش کمتری برخودار است. در واقع این روشها حجم عظیمی از روابط را بدون دادن دانشی در زمینه نوع آرگومان های آن یا نوع خود رابطه و جایگاه آن نسبت به سایر روابط تولید میکنند. بکار گرفتن این اطلاعات در کاربردهایی مانند درک متن و مخصوصا استلزام از متن مشکل است. در ادامه چندین مقاله مشهور در هر دو زمینه استخراج رابطه آزاد و غیر آزاد را بررسی میکنیم.

**استخراج رابطه**

مقالات دانشگاه استنفورد

**استخراج رابطه آزاد**

در این دسته از الگوریتمها، هدف استخراج هر گونه دانش موجود در متن است. از قبل هدف مشخص یا پایگاه داده روابط را در اختیار نداریم. آزمایشگاه پردازش متن دانشگاه واشنگتن[[59]](#footnote-59) یکی از فعال ترین آزمایشگاه ها در این زمینه است. پروژه معروف آنها با نام KnowItAll نیز در زمینه استخراج اطلاعات آزاد از متن است.

**معرفی الگوریتم TextRunner** (Banko, 2007)

این الگوریتم اولین روش معرفی شده در زمینه استخراج رابطه آزاد است. در واقع زمینه استخراج اطلاعات آزاد اولین بار با این مقاله در سال 2007 معرفی گردید. تا قبل از آن در تمام مقالات لیست روابط هدف از قبل در اختیار الگوریتم قرار می­گرفت. مثلا الگوریتمی ارائه می­شد تا لیست تمام کتاب های یک نویسنده را استخراج کند. در ادامه مراحل اجرای الگوریتم به صورت خلاصه ارائه می­گردد.

1. ابتدا با استفاده از یک پارسر چند صد جمله را بررسی کرده و از داخل آنها مجموعه ای از روابط را استخراج می­کند. این روابط در صورت درست بودن برجسب Positive و در صورت غلط بودن برچسب Negative را میخورند.
2. برای تعیین درست یا غلط بودن هر رابطه، یکسری محدودیت نحوی که عموما روی زنجیره وابستگی Dependency chain جمله کار میکند تعریف کرده است.
3. یک مجموعه ویژگی که محاسبه آنها نیازی به پارسر ندارد تعریف کرده است. برای تمام روابط استخراج شده اعمم از درست یا غلط این ویژگی ها را محاسبه میکنیم.
4. بردارهای ویژگی محاسبه شده را به همراه برچسب شان به یک دسته بندی کننده Naïve Bayes می­دهیم.
5. از این دسته بندی کننده برای استخراج روابط جدید استفاده میکنیم.

**معرفی الگوریتم ReVerb** (Anthony Fader, 2011)

در ادامه معرفی الگوریتم های جدید در پروژه KnowItAll این الگوریتم در سال 2011 توسط دانشجویان دانشگاه واشنگتن معرفی شد. در این مقاله حل دو مشکل سیستم های قبلی (مخصوصا TextRunner) مورد توجه قرار گرفته است.

1. استخراج های بی ربط (نا مفهوم)[[60]](#footnote-60): زمانی که اطلاعات استخراج شده بی معنا هستند. در شکل زیر نمونه هایی از این دست اطلاعات را مشاهده می­کنید.

****

تصویر 11 نمونه هایی از استخراج بی ربط. این گونه خروجی ها حدود 13 درصد خروجی های الگوریتم TextRunner و 15 درصد خروجی های الگوریتم WOE را شامل میگردد.

1. استخراج های بی ارزش[[61]](#footnote-61): زمانی که در فرایند استخراج، اطلاعات حیاتی جا می­افتند. مثلا برای جمله «Jack claimed responsibility for Anne attack» رابطه زیر استخراج شده است. (Jack, claimed, responsibility). در حالیکه استخراج درست (Jack, claimed responsibility for, Anne attack) است. این گونه خطاها معمولا به خاطر ناتوانی در پردازش درست افعال (LVC[[62]](#footnote-62)) است. این افعال دارای دو بخش فعل و اسم هستند که بخش اسم بار معنایی جمله را بدوش می­کشد. بنابراین حذف بخش اسم موجب بی ارزش شدن رابطه می­گردد. در تصویر 12 نمونه هایی از استخراج های بی ارزش را مشاهده میکنید.



تصویر 12 نمونه هایی از استخراج اطلاعات بی ارزش (ستون چپ) و نمونه کامل شده آنها (ستون راست). اطلاعات بی ارزش 4 درصد از خروجی های الگوریتم WOE و 7 درصد از خروجی های الگوریتم TextRunner را شامل میشود.

این روش برای حل دو مشکل مطرح شده است. محدودیت اصلی الگوریتم ReVerb همانطور که از نامش پیدا است این است که تنها روابطی را که فعل هستند استخراج می­کند. میدانیم که اطلاعات می­توانند در گروه های اسمی و حتی قیدی متن قرار گرفته باشند. مثلا از گروه اسمی ابتدای جمله «حسن روحانی رئیس جمهوری ایران در ملاقات با همتای کره ای خود گفت» میتوان نتیجه گرفت که (حسن روحانی، رئیس جمهوری، ایران) این الگوریتم از استخراج اینگونه اطلاعات از متن عاجز است. در ادامه خواهیم دید که در سال 2012 با ارائه الگوریتم OLLIE سعی در حل این مشکل نمودند.

برای حل این دو مشکل دو محدودیت در این مقاله معرفی شده اند.

1. محدودیت نحوی[[63]](#footnote-63): اولا استخراج های بی ربط را حذف میکند. ثانیا استخراج های بی ارزش را کاهش می­دهد. محدودیت نحوی عبارت رابطه را مجبور میکند که از الگوی POS مطرح شده پیروی کند. در این الگو رابطه یا یک فعل ساده مثل (invented) است یا یک عبارت فعلی که بلافاصله بعد از آن حرف اضافه میآید مانند (located in) یا اینکه عبارت فعلی که بعد از آن عبارت اسمی میاید و با یک حرف اضافه خاتمه میابد. مثلا (has atomic weight of) . در هنگام اعمال محدودیت نحوی دو شرط زیر را رعایت میکنیم:
   1. اگر چندین تطابق در یک جمله برای فعل داشته باشیم بزرگترین را انتخاب میکنیم.
   2. اگر چندین دنباله تطابق متوالی داشتیم آنها را با هم ترکیب میکنیم. مثلا (wants to extend)

در تصویر 13 مجموعه محدودیت نحوی مورد استفاده در الگوریتم ReVerb را مشاهده می­کنید



تصویر 13 عبارات منظم مبتنی بر برچسب مقوله نحوی مورد استفاده در الگوریتم ReVerb برای کاهش اتخراج های بی ربط

1. محدودیت لغوی[[64]](#footnote-64): برخی از خروجی های محدودیت نحوی آنقدر خاص هستند که در گستره ای به اندازه وب نیز فقط چند بار رخ می­دهند. مثلا در جمله (the Obama administration is offering only modest greenhouse gas reduction targets at the conference.) محدودیت نحوی با عبارت زیر تطابق میابد (is offering only modest greenhouse gas reduction targets at) که این رابطه خوب نیست زیرا بسیار خاص شده است. محدودیت لغوی می­گوید یک رابطه در صورتی مناسب است که با آرگومانهای متفاوت (حداقل k=20 آرگومان) در یک دادگان بزرگ (مثلا در وب) آمده باشد.

برای پیاده سازی محدودیت لغوی از یک دیکشنری بزرگ (D) از عبارات رابطه که می­توانند آرگومانهای مختلفی بگیرند استفاده میکند. برای ساخت این دیکشنری از محدودیت نحوی تعریف شده برای تولید عبارات رابطه در یک دادگان از وب با 500 میلیون جمله استفاده شده، به صورت اکتشافی[[65]](#footnote-65) آرگومانها را یافته، عبارات روابط را فیلتر کرده است. این فیلترینگ به صورتی است که هر عبارتی که کمتر از k<20 آرگومان مختلف دارد را حذف کرده است.

مزایای این روش به شرح زیر است:

1. عبارت رابطه به صورت کلی یافته میشود نه کلمه به کلمه. این مزیتی است نسبت به روشهای قبلی مبتنی بر برچسب زدن توالی که از مدل های گرافیکی آماری مانند میدان شرطی تصادفی استفاده می­کردند. (Etzioni M. B., 2008) مشکل این روشها در این بود که چون عبارت رابطه را به صورت کلمه به کلمه می­یافتند، معمولا روابط تولید شده چند کلمه زیاد یا کم داشت.
2. عبارات دارای پتانسیل رابطه شدن، با استفاده از آمار روی داده های بزرگ فیلتر شده اند. (پیاده سازی محدودیت لغوی)
3. این روش در ابتدا رابطه را می­یابد بعد آرگومان ها را. اینکار جلو اشتباه گرفتن یک اسم در عبارت رابطه را به عنوان آرگومان می­گیرد. این مشکل در الگوریتم های قبلی زیاد دیده می­شد.

الگوریتم تشخیص رابطه در ادامه ارائه شده است.

ورودی الگوریتم یک جمله برچسب مقوله نحوی خورده و گروه های اسمی تعیین شده است. خروجی مجموعه ای از روابط به فرم (x,r,y) است که در آن x و y آرگومان های رابطه و r خود رابطه است.

1. ابتدا با روش مطرح شده برای هر فعل، رابطه را با توجه به دو محدودیت نحوی و لغوی تعیین می­کنیم. (پارامتر r)
2. نزدیکترین گروه اسمی سمت چپ رابطه (r) که ضمیر نسبی یا WHO-adverb یا «there» وجودی نیست را می­یابیم. (آرگومان اول)
3. نزدیکترین گروه اسمی سمت راست رابطه را هم میابیم. (آرگومان دوم)
4. اگر این دو آرگومان را بیابیم رابطه (x,r,y) را در خروجی تولید می­کنیم.

**معرفی الگوریتم R2A2 (Etzioni O. a., 2011):**

در این مقاله ابتدا خروجی الگوریتم ReVerb بررسی شده است. در آزمایشاتی که ترتیب دادند متوجه شدند که در 65 درصد خطاهای ReVerb، رابطه به درستی تعیین شده است ولی خطا در تعیین آرگومانها رخ داده. زیرا روش اکتشافی مورد استفاده در آن برای تعیین آرگومانها چندان قدرتمند نیست. یک بخش عمده از بقیه خطاها هم در اشتباه گرفتن یک رابطه n-آرگومانی با یک رابطه دو آرگومانی بوده است. مثلا از جمله «I gave him 15 photographs» به غلط رابطه (I, gave, him) استخراج شده است. همچنین در 52 درصد از موارد منفی کاذب[[66]](#footnote-66) عدم شناسایی رابطه بدلیل ناتوانی در تعیین آرگومانهای آن بوده است. مثلا در جمله «The cost of the war against Iraq has risen above 500 billion dollars» رابطه زیر تولید می­شود «(Iraq, has risen above, 500 billion dollars)» و از جمله «The plan would reduce the number of teenagers who begin smoking» رابطه (The plan, would reduce the number of, teenagers) که درمورد اول در شناسایی آرگومان اول و در مورد دوم در شناسایی آرگومان دوم اشتباه شده است.

برای حل این مشکلات یک سیستم مخصوص شناسایی آرگومان های رابطه با نام ArgLearner طراحی شده است تا جایگزین روش اکتشافی قبلی گردد. در این روش برای شناسایی آرگومان های فعل از سه دسته بندی کننده استفاده شده است. (دو تا برای تعیین شروع و پایان آرگومان اول، و یکی برای شناسایی پایان آرگومان دوم) ادعا شده است که همیشه در زبان انگلیسی شروع آرگومان دوم بلافاصله بعد از اتمام رابطه است و بنابراین نیازی به تعیین مکان شروع آرگومان دوم نداریم. برای پیاده سازی دسته بندی کننده ها از Weka REPTree پیاده سازی مقاله (Hall, 2009) و میدان تصادفی شرطی پیاده سازی شده در مجموعه Mallet (McCallum, 2002) استفاده شده است. به سیستم ReVerb ای که شناسایی آرگومان آن با روش ArgLearner انجام گیرد R2A2 گوییم.

در تصویر 14 معماری الگوریتم R2A2 را مشاهده می­کنید.



تصویر 14 معماری الگوریتم R2A2 و سیستم ArgLearner

در تصویر 15 مقایسه عملکرد دو الگوریتم R2A2 و ReVerb را مشاهده می­کنید.



تصویر 15 الگوریتم R2A2 دقت و فراخوان بسیار بالاتری نسبت به ReVerber دارد

همانطور که می­بینید معیار دقت و فراخوان به شکل چشمگیری بهبود یافته است.

**معرفی الگوریتم اوولی[[67]](#footnote-67)** (Schmitz M. a., 2012)

این الگوریتم در سال 2012 معرفی گردید. هدف آن بهبود عملکرد الگوریتم ReVerb (Anthony Fader, 2011) بود. در آن سال لبه تکنولوژی سیستم های استخراج اطلاعات آزاد الگوریتم ReVerb بود که دو مشکل عمده زیر را داشت.

1. تنها می­توانست روابط موجود در افعال را استخراج کند.
2. به زمینه[[68]](#footnote-68) متن توجه نمیکرد. این امر باعث می­شد روابطی که به عنوان حقیقت در متن ذکر نشده اند را به غلط استخراج کند

در الگوریتم اوولی سعی شده است این دو مشکل مرتفع گردند. این روش با استخراج روابط موجود در گروه های اسمی و صفت موصوفی ها بازده[[69]](#footnote-69) را بالا برده است. همچنین با بکارگیری یک تحلیلگر زمینه متن توانسته است دقت سیستم را بالا ببرد. این تحلیلگر در صورت نیاز اطلاعات زمینه ای متن را به مسند استخراج شده پیوست می­کند. اوولی توانست مساحت زیر نمودار دقت-بازدهی[[70]](#footnote-70) را 2.7 برابر بهتر از ReVerb کند. در جدول زیر مقایسه ای بین خروجی الگوریتم اوولی با دو الگوریتم ReVerb و WOE (Weld, 2010) را می­بینید.

|  |
| --- |
| 1. “After winning the Super bowl, the Saints are now the top dogs of the NFL.”  O: (the Saints; win; the Super bowl)  2. “There are plenty of taxis available at Bali airport.”  O: (taxis; be available at; Bali airport)  3. “Microsoft co-founder Bill Gates spoke at ...”  O: (Bill Gates; be co-founder of; Microsoft) |
| 4. “Early astronomers believed that the earth is the center of the universe.”  R: (the earth; be the center of; the universe)  W: (the earth; be; the center of the universe)  O: ((the earth; be the center of; the universe) AttributedTo believe; Early astronomers)  5. “If he wins five key states, Romney will be elected President.”  R,W: (Romney; will be elected; President)  O: ((Romney; will be elected; President) ClausalModifier if; he wins five key states) |

جدول 1خروجی الگوریتم اوولی با علامت O، الگوریتم WOE با علامت W و الگوریتم ReVerb با علامت R نشان داده شده اند.

همانطور که در جدول 1 مشاهده می­کنید، برای سه مثال اول تنها اوولی توانسته است خروجی مناسب را تولید کند. زیرا اطلاعات موجود در این سه مثال در فعل آنها قرار نگرفته اند. در مثال 4 نیز جمله در حال ارائه حقیقت نیست بلکه تنها عقاید گذشتگان را مطرح می­کند. میدانیم که زمین مرکز عالم نیست. در جمله آخر نیز تنها یک حالت شرطی بیان شده است و از نتیجه شرط اطلاعی نداریم بنابراین خروجی های دو سیستم ReVerb و WOE غلط بوده و تنها اوولی درست کار کرده است.

در شکل زیر معماری سیستم اوولی را مشاهده می­کنید.



تصویر 16 معماری سیستم اوولی. شروع با دانه های تولید شده با روش ReVerb است. از این دانه ها برای تولید مجموعه آموزشی روش خود راه انداز بهره میبرد. الگوهای باز را یاد میگیرد که در زمان استخراج روی جملات آزمایشی اعمال میشوند.

این روش در مرحله اول از یک مجموعه دانه تولید شده با روش ReVerb که دارای دقت بالایی هستند به عنوان ورودی یک سیستم خود راه انداز[[71]](#footnote-71) استفاده می­کند. روش خود راه انداز مبتنی بر یادگیری نیمه نظارتی[[72]](#footnote-72) است. در این روش مجموعه ای معمولا کوچک از داده های برچسب خورده داریم. از این داده ها برای برچسب زدن نمونه های موجود و تولید یک پایگاه داده بزرگ آموزشی بهره گرفته می­شود. سپس سیستم روی این پایگاه داده آموزش میبیند. در مرحله دوم الگوریتم اوولی مجموعه از الگوهای آزاد[[73]](#footnote-73) را از روی پایگاه داده آموزش تولید میکند. سپس از این الگوهای آزاد در زمان استخراج روابط بهره میگیرد. در انتها زمینه اطراف متن را برای اضافه کردن اطلاعات زمینه ای (AttributedTo، ClausalModifier) به مسند های استخراج شده بررسی میکند.

با 110،000 دانه با معیار اطمینان بالای تولید شده از روش ReVerb آغاز میکند. به عنوان نمونه یک دانه تولید شده (Paul Annacone, is the coach of, Federer) از روی جمله «Paul Annacone is the coach of Federer» است. برای هر دانه روی پایگاه داده آنلاین [[74]](#footnote-74)(ClueWeb) حرکت کرده و تمام جملات دارای محتوای آن را استخراج میکنیم. در مجموع 18 میلیون جمله حاصل شده است. به عنوان نمونه برای دانه بالا جمله زیر استخراج شده است. «Now coached by Annacone, Federer is winning more titles than ever.» فرض الگوریتم خود راه انداز این است که تمام این جملات، اطلاعات جمله اصلی را بیان میکنند. این فرض همیشه درست نیست. به عنوان مثال ممکن است برای دانه (Boyle; is born in; Ireland) جمله روبرو استخراج شود. «Felix G. Wharton was born in Donegal, in the north west of Ireland, a country where the Boyles did their schooling» که اطلاعات دانه اصلی را ندارد. برای کاهش این خطاها از پارسر وابستگی و محدودیت گذاشتن روی فاصله بین عناصر رابطه استفاده شده است. پارسر مورد استفاده هم برای زبان انگلیسی بوده و Malt Dependency Parser (Nivre, 2004) نام دارد. روش خود راه انداز در مقالات پیشین نیز برای تولید نمونه مثبت مورد استفاده بوده است (Raphael Hoffmann, 2010). اما مقالات پیشین جملاتی را استخراج میکردند که تنها محتوی دو آرگومان رابطه باشند که این کار خطا خیز است زیرا مثلا «بیل گیتس» میتواند روابط «مدیر عامل»، «موسس» یا «سهام دار» را با «شرکت مایکروسافت» داشته باشد.

در مرحله بعد اوولی از روی داده های حجیم آموزشی تولید شده، الگوهای آزاد را تولید میکند. در زیر نمونه هایی از الگوهای آزاد استخراج شده را مشاهده میکنید.



تصویر 17 نمونه الگوهای آزاد تولید شده با الگوریتم اوولی. توجه کنید که برخی از الگوها {1و2و3} کاملا نحوی و برخی {4و5} نحوی با محدودیت معنایی-لغوی هستند.

برای تولید این الگوها از پارسر وابستگی معرفی شده استفاده شده است. ابتدا در داخل درخت پارس، با توجه به برچسب های نحوی، عبارات محتمل برای رابطه شناسایی شده است. سپس ساختار کلی رابطه از روی درخت تولید شده است. ابتدا الگوها از نوع نحوی {الگوهای 1و2و3} ساخته میشوند. الگوی نحوی قویترین نوع الگو است زیرا محدودیتی ندارد. اگر در ادامه بررسی دادگان متوجه وجود خطا در استخراج یک الگوی نحوی شویم آن را یک مرحله پایین آورده {الگوی 4} و سعی میکنیم مشکل را با محدودیت Type Generalization حل کنیم. مثلا متوجه میشویم که یک رابطه فقط زمانی درست است که آرگومان اول آن یک Person باشد. البته این مقاله کار Type Generalization را فقط برای دو نوع Person و Location انجام میدهد. اگر با اینکار باز هم مواردی دیده شده که الگو استخراج غلط انجام میدهد، آن را یک مرحله پایین تر آورده {الگوی 5} و محدودیت لغوی نیز به آن الگو اضافه میکنیم. مثلا متوجه میشویم یک رابطه تنها برای یکسری افعال مشخص برقرار است و نباید به همه افعال عمومیت داده شود.

برای شناسایی الگو در داده های آزمایشی، باز هم با استفاده از درخت وابستگی الگوهای آزاد را با این داده ها تطابق میدهیم. در شکل زیر نمونه ای از استفاده از درخت وابستگی برای استخراج رابطه از جمله را مشاهده میکنید.



تصویر 18 نمونه ای از درخت وابستگی. گره های خاکستری رنگ با استفاده از الگوی

{arg1} ↑ nsubjpass ↑ {rel:postag=VBN} ↓ {prep\_\*} ↓ {arg2}

تولید شده اند. رابطه تولید شده (the 2012 Sasquatch Music Festival; is scheduled for; May 25th)

در شکل زیر مقایسه این روش را با دو روش لبه تکنولوژی دیگر مشاهده می­کنید. میتوان گفت در زمینه استخراج اطلاعات آزاد این الگوریتم بهترین نتایج را داشته است.



تصویر 19 روش اوولی مساحت بیشتری زیر نمودار بازده-دقت دارد

1. **ایده ها و روش های پیشنهادی**

1. Information extraction (IE) [↑](#footnote-ref-1)
2. Open information extraction (Open IE) [↑](#footnote-ref-2)
3. Part of speech tags (POS) [↑](#footnote-ref-3)
4. Clear factual information [↑](#footnote-ref-4)
5. Selectional preferences [↑](#footnote-ref-5)
6. Common sense knowledge [↑](#footnote-ref-6)
7. Textual entailment [↑](#footnote-ref-7)
8. Ontology [↑](#footnote-ref-8)
9. Named entity recognition (NER) [↑](#footnote-ref-9)
10. Entity [↑](#footnote-ref-10)
11. Sentiment analysis (opinion mining) [↑](#footnote-ref-11)
12. Precision recall [↑](#footnote-ref-12)
13. Tagger [↑](#footnote-ref-13)
14. MUC score [↑](#footnote-ref-14)
15. F1 measure [↑](#footnote-ref-15)
16. Discriminative estimation [↑](#footnote-ref-16)
17. Generative models [↑](#footnote-ref-17)
18. Naïve Bayes [↑](#footnote-ref-18)
19. Language models [↑](#footnote-ref-19)
20. Conditionals discriminative models [↑](#footnote-ref-20)
21. Joint distribution [↑](#footnote-ref-21)
22. N-gram model [↑](#footnote-ref-22)
23. Hidden Markov model [↑](#footnote-ref-23)
24. Logistic regression [↑](#footnote-ref-24)
25. Support vector machine [↑](#footnote-ref-25)
26. Probabilistic graphical models [↑](#footnote-ref-26)
27. Word sense disambiguation [↑](#footnote-ref-27)
28. Smoothing [↑](#footnote-ref-28)
29. Overfitting [↑](#footnote-ref-29)
30. Feature [↑](#footnote-ref-30)
31. Expected value [↑](#footnote-ref-31)
32. Matching predicate [↑](#footnote-ref-32)
33. Text categorization [↑](#footnote-ref-33)
34. Bag of words [↑](#footnote-ref-34)
35. Prepositional phrases [↑](#footnote-ref-35)
36. Gradient descent [↑](#footnote-ref-36)
37. LBFGS [↑](#footnote-ref-37)
38. Log-linear classifier [↑](#footnote-ref-38)
39. Gibbs [↑](#footnote-ref-39)
40. Sequential model [↑](#footnote-ref-40)
41. Character sub sequences [↑](#footnote-ref-41)
42. Word shape sequences [↑](#footnote-ref-42)
43. Maximum entropy sequence model [↑](#footnote-ref-43)
44. Semantic role labeling [↑](#footnote-ref-44)
45. Tokenization [↑](#footnote-ref-45)
46. Maximum entropy Markov models [↑](#footnote-ref-46)
47. Greedy [↑](#footnote-ref-47)
48. Beam inference [↑](#footnote-ref-48)
49. Viterbi inference [↑](#footnote-ref-49)
50. Dynamic programming [↑](#footnote-ref-50)
51. Conditional random fields [↑](#footnote-ref-51)
52. Conditional likelihood [↑](#footnote-ref-52)
53. State of the art [↑](#footnote-ref-53)
54. Open information extraction [↑](#footnote-ref-54)
55. Word net [↑](#footnote-ref-55)
56. Frame net [↑](#footnote-ref-56)
57. Relation extraction [↑](#footnote-ref-57)
58. Entity tuple [↑](#footnote-ref-58)
59. <http://openie.cs.washington.edu/> [↑](#footnote-ref-59)
60. Incoherent extraction [↑](#footnote-ref-60)
61. Uninformative extractions [↑](#footnote-ref-61)
62. Light verb construction [↑](#footnote-ref-62)
63. Syntactic constraint [↑](#footnote-ref-63)
64. Lexical constraint [↑](#footnote-ref-64)
65. Heuristic [↑](#footnote-ref-65)
66. False negative [↑](#footnote-ref-66)
67. OLLIE [↑](#footnote-ref-67)
68. Context [↑](#footnote-ref-68)
69. Yield [↑](#footnote-ref-69)
70. AUC curve [↑](#footnote-ref-70)
71. Bootstrapping [↑](#footnote-ref-71)
72. Semi supervised learning [↑](#footnote-ref-72)
73. Open pattern [↑](#footnote-ref-73)
74. <http://lemurproject.org/clueweb09.php/> [↑](#footnote-ref-74)